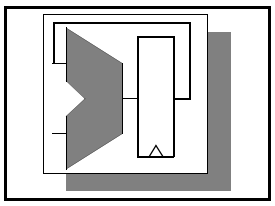
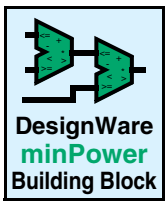




DW_div_seq

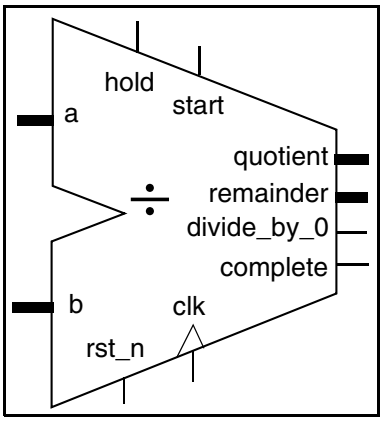
Sequential Divider

Version, STAR, and myDesignWare Subscriptions: IP Directory

Features and Benefits

Revision History

- Parameterized word length
- Parameterized number of clock cycles
- Unsigned and signed (two's complement) data division
- Registered or un-registered inputs and outputs
- Includes a low-power implementation (at a sub-level) that has power benefits from minPower optimization (for details, see Table 1-3 on page 3)



Description

DW_div_seq is a sequential divider designed for low area, area-time trade-off, or high frequency (small cycle time) applications. DW_div_seq is an integer divider with both `quotient` and `remainder` outputs.

Table 1-1 describes the pins for DW_div_seq.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
clk	1 bit	Input	Clock
rst_n	1 bit	Input	Reset, active low
hold	1 bit	Input	Hold current operation (=1)
start	1 bit	Input	Start operation (=1) A new operation is started by setting `start` =1 for one clock cycle.
a	*a_width* bits	Input	Dividend
b	*b_width* bits	Input	Divisor
complete	1 bit	Output	Operation completed (=1)
divide_by_0	1 bit	Output	Indicates if `b` equals 0
quotient	*a_width* bits	Output	Quotient
remainder	*b_width* bits	Output	Remainder

Table 1-2 describes the parameters for DW_div_seq.

Table 1-2 Parameter Description

Parameter	Values	Description
a_width	≥ 3 Default 8	Word length of `a`
b_width	3 to *a_width* Default: 8	Word length of `b`
tc_mode	0 or 1 Default: 0	Two's complement control ■ 0 = Unsigned ■ 1 = Two's complement
num_cyc	3 to *a_width* Default: 3	User-defined number of clock cycles to produce a valid result. The real number of clock cycles depends on various parameters and is given in Table 1-6 on page 4 and the topic titled "Formula for Dividend Bits Processed Per Cycle" on page 5.
rst_mode	0 or 1 Default: 0	Reset mode ■ 0 = Asynchronous reset ■ 1 = Synchronous reset
input_mode[a]	0 or 1 Default: 1	Registered inputs ■ 0 = No ■ 1 = Yes
output_mode	0 or 1 Default: 1	Registered outputs ■ 0 = No ■ 1 = Yes
early_start	0 or 1 Default: 0	Computation start ■ 0 = Start computation in the second cycle ■ 1 = Start computation in the first cycle For the dependency of *early_start* on *input_mode*, see Table 1-6 on page 4.

a. When configured with the parameter *input_mode* set to '0', the inputs `a` and `b` MUST be held constant from the time `start` is asserted until `complete` has gone high to signal completion of the calculation. Conversely, if a configuration with the parameter *input_mode* set to '1' is used, the `a` and `b` inputs will be captured when `start` is high and otherwise ignored.

Table 1-3 Synthesis Implementations

Implementation	Function	License Feature Required
cpa[a]	radix-2 synthesis model	DesignWare[b]
cpa2[a]	radix-4 synthesis model	DesignWare[b]

a. To achieve low-power benefits in sub-module implementations, you need to enable minPower; for details, see "Enabling minPower" on page 9.
b. For releases prior to P-2019.03, the DesignWare-LP license feature is required to achieve low-power benefits.

Table 1-4 Simulation Models

Model[a]	Function
DW03.DW_DIV_SEQ_CFG_SIM	Design unit name for VHDL simulation
dw/dw03/src/DW_div_seq_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_div_seq.v	Verilog simulation model source code

a. Note that during the computation phase (after start and before complete is asserted), the simulation models output X values and therefore cannot be used as a compare for gate-level simulations.

Attention

A divide by zero warning message is generated each time the `b` input changes to the value 0. This warning can be disabled for Verilog simulations by defining the Verilog macro, DW_SUPPRESS_WARN, either in the test bench or on the simulator command line (such as, `+define+DW_SUPPRESS_WARN+`).

Table 1-5 Operation Truth Table

start	hold	Operation
0	0	Idle or Running
0	1	Hold
1	X	Start

This component divides the dividend `a` by the divisor `b` to produce the `quotient` and `remainder` in a user-defined number of clock cycles (*num_cyc*). As long as `start`=1 the division operation is in the initialization state. Once `start`=0, the calculation begins followed by valid output flagged when `complete` =1 or an intervening setting of `start` to 1. The divide operation is stalled when `hold` =1. For a description of divide operation, refer to the datasheet DW_div. DW_div_seq calculates the division remainder (not modulus) on the output `remainder`, which corresponds to the "rem" operator in VHDL and the "%" operator in Verilog. This is equivalent to the `remainder` output of DW_div with *rem_mode=1*.

The parameter `tc_mode` determines whether the data of inputs (`a, b`) and outputs (`quotient, remainder`) are interpreted as unsigned (*tc_mode = 0*) or two's complement (*tc_mode =1*) numbers.

The internal registers can either have an asynchronous (*rst_mode* = 0) or synchronous reset (*rst_mode =1*) that is connected to the reset signal `rst_n`.

After reset conditions are released (`rst_n` =1) there are no restrictions on when `start` can be set to 1 and then to 0. However, if `start` is set to 0 immediately after `rst_n` goes to 1, and `start`=0 continues through the first *num_cyc* clock cycles, then `complete` will go to 1. This first `complete` =1 when no start is initiated following reset may yield invalid results and should be disregarded.

The parameter *input_mode* determines whether the inputs are registered inside DW_div_seq (*input_mode=1*) or not (*input_mode=0*). If configured without input registers (*input_mode=0*), the logic that drives the inputs `a` and `b` must hold the input values constant for the entire time it takes to calculate the result (from the cycle before `start` drops until `complete` goes high). When configured with input registers (*input_mode=1*) the inputs `a` and `b` are captured when `start` is high and ignored until `start` goes high again.

When configured with no input registers, changes on inputs `a` and `b` while `complete` is low (calculation cycle) produces unpredictable output values. Simulation models will produce unknown output values (Xs) and post an error message indicating the instance that violated this rule and the simulation time when the violation was detected.

The parameter *output_mode* determines whether the outputs are registered.

When the parameter *early_start* is set to 1, computation starts using the value at input `b` on the first cycle (the last cycle before `start` goes to 0), and switching to the registered `b` input on the following cycles. This saves one extra cycle to store the data (*early_start=0*), but feeds the inputs directly into the components critical path.

Table 1-6 shows the *input_mode*, *output_mode*, and *early_start* parameter combinations and corresponding actual number of cycles required to perform an operation.

Table 1-6 Actual Cycles Based on *input_mode*, *output_mode*, and *early_start*

input_mode	*output_mode*	*early_start*	Actual Number of Cycles
0	0	0	*num_cyc*–2
0	0	1	Invalid parameter setting
0	1	0	*num_cyc*–1
0	1	1	Invalid parameter setting
1	0	0	*num_cyc*–1
1	0	1	*num_cyc*–2
1	1	0	*num_cyc*
1	1	1	*num_cyc*–1

Note that the *num_cyc* value indicates the actual throughput of the device from when `start` is asserted to when `complete` is asserted. However, if a calculation is in progress (before the *num_cyc* number of cycles has been reached) when `start` is asserted again, the results are undetermined until `complete` is asserted. The results associated with the assertion of `complete` are from the input values from the previous assertion of `start`.

Formula for Dividend Bits Processed Per Cycle

The following formula describes the number of dividend bits processed per cycle:

bits processed per cycle = ceil (*a_width*/*num_cyc*)

where:

a_width is the bit width of the dividend (as defined in Table 1-2 on page 2)

num_cyc is the number of clock cycles required for division (as defined in Table 1-2 on page 2)

Formula for Actual Number of Cycles Required

The actual number of clock cycles required for a computation is calculated using the following formula:

actual number of cycles required = *num_cyc* - (1 - *output_mode*) - (1 - *input_mode*) - *early_start*

where:

num_cyc is the number of clock cycles required for division (as defined in Table 1-2 on page 2)

output_mode is the control for registered output (as defined in Table 1-2 on page 2)

input_mode is the control for registered inputs (as defined in Table 1-2 on page 2)

early_start is the control for when the computation starts (as defined in Table 1-2 on page 2)

Suppressing Warning Messages During Verilog Simulation

The Verilog simulation model includes macros that allow you to suppress warning messages during simulation.

To suppress all warning messages for all DWBB components, define the DW_SUPPRESS_WARN macro in either of the following ways:

- Specify the Verilog preprocessing macro in Verilog code:

  ```
  `define DW_SUPPRESS_WARN
  ```

- Or, include a command line option to the simulator, such as:

  `+define+DW_SUPPRESS_WARN` (which is used for the Synopsys VCS simulator)

The warning messages for this model include the following:

- If values other than 1 or 0 are present on a clock port, the following message is displayed:

  ```
  WARNING: <instance_path>.<clock_name>_monitor:
      at time = <timestamp>, Detected unknown value, x, on <clock_name> input.
  ```

  To suppress only this warning message for all DWBB components, use the following macro:

  - Define the DW_DISABLE_CLK_MONITOR macro. You can define this macro in the following ways:
    - Specify the Verilog preprocessing macro in Verilog code:

      ```
      `define DW_DISABLE_CLK_MONITOR
      ```

    - Or, include a command line option to the simulator, such as:

      `+define+DW_DISABLE_CLK_MONITOR` (which is used for the Synopsys VCS simulator)

  This message is also suppressed using the DW_SUPPRESS_WARN macro explained earlier.

- If a divide-by-zero operation is attempted, the following message is displayed:

  ```
  WARNING: <instance_path>:
      at time = <timestamp>: Division by zero.
  ```

  To suppress this message, use the DW_SUPPRESS_WARN macro explained earlier.

- If the component is configured without an input register and an input operand changes during calculation, the following message is displayed:

  ```
  WARNING: <instance_path>:
      at time = <timestamp>, Operand input change on DW_div_seq during calculation
  (configured without an input register) will cause corrupted results if operation is
  allowed to complete.
  ```

  To suppress this message, use the DW_SUPPRESS_WARN macro explained earlier.

- If the component is configured without input and output registers and an input operand changes during calculation, the following message is displayed:

  ```
  WARNING: <instance_path>:
      at time = <timestamp>, Operand input change on DW_div_seq during calculation
  (configured with neither input nor output register) causes output to no longer retain
  result of previous operation.
  ```

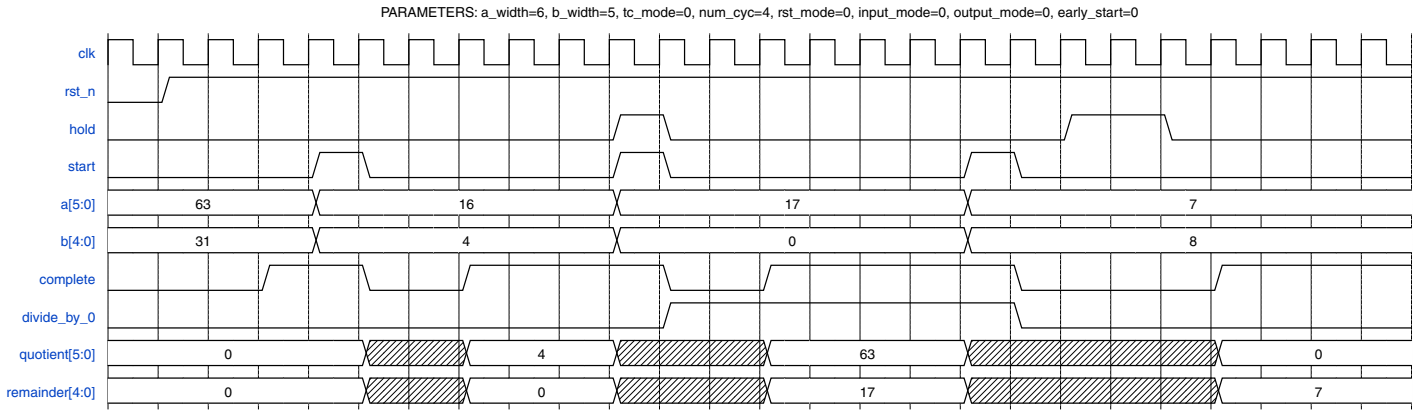
  To suppress this message, use the DW_SUPPRESS_WARN macro explained earlier.

Timing Waveforms

The following timing waveforms show a 6-bit by 5-bit unsigned sequential divider for specific inputs of `hold` and `start` and their corresponding outputs. The parameter settings for each simulation are shown at the top of each figure. When `hold` = 1 and `start` = 0, the result is delayed by the same number of clock cycles for which `hold` is 1. For example, if `hold`=1 for two clock cycles, then the result is delayed by two clock cycles.

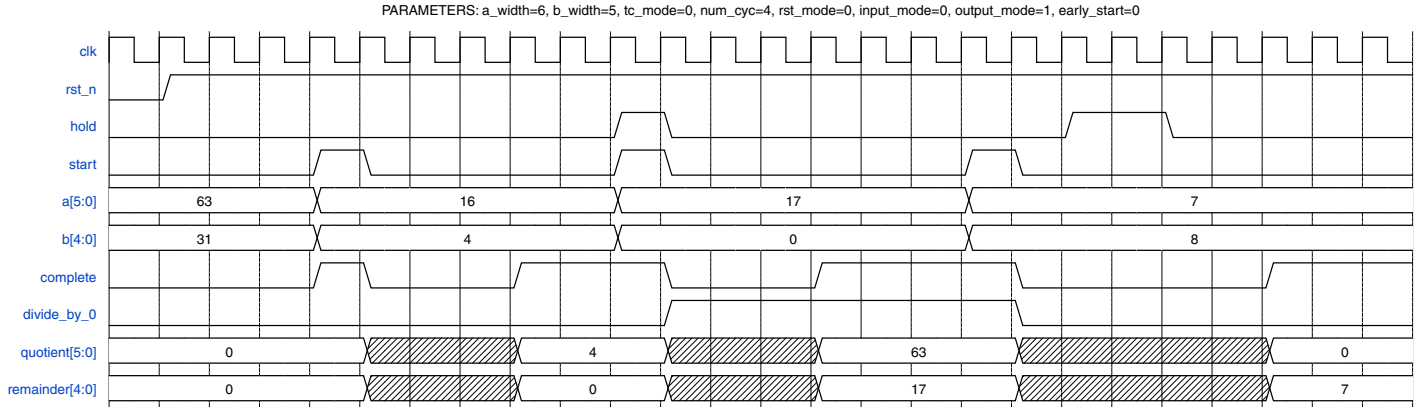
For the parameter settings shown in Figure 1-1 on page 7, Table 1-6 on page 4 specifies that the result is produced 2 clock cycles after `start` is set to 0, as shown in Figure 1-1 on page 7 for the operations where `hold` is kept low. For this set of parameters, the result must be read before a new input is applied, given that there is no input or output registers. Driving a `start` signal or changing the input values (dividend or divisor) before reading the output results in undetermined data.

Figure 1-1 Simulation Waveform 1



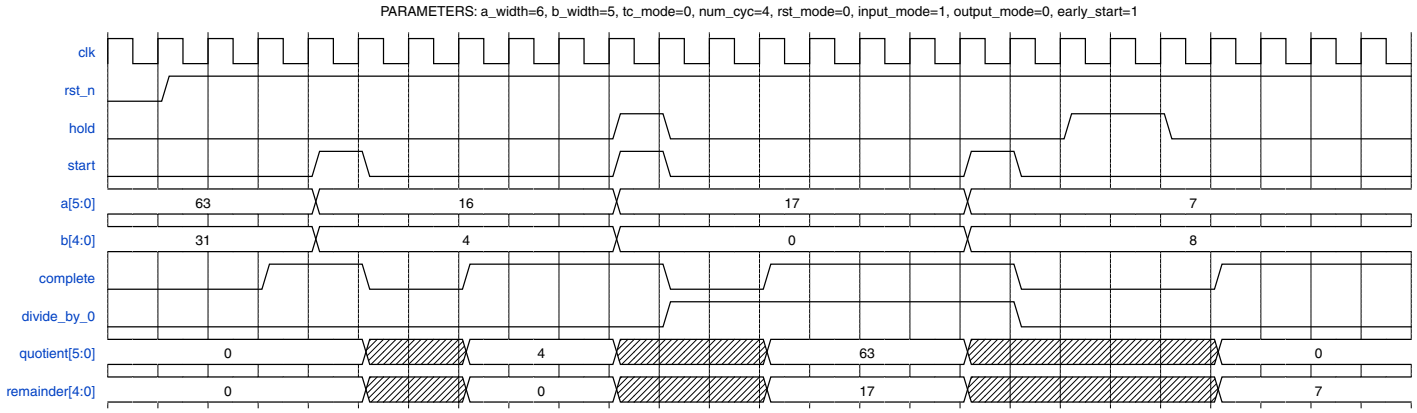
For the parameter settings shown in Figure 1-2, Table 1-6 on page 4 specifies that the result is produced three cycles after `start` goes to 0.

Figure 1-2 Simulation Waveform 2



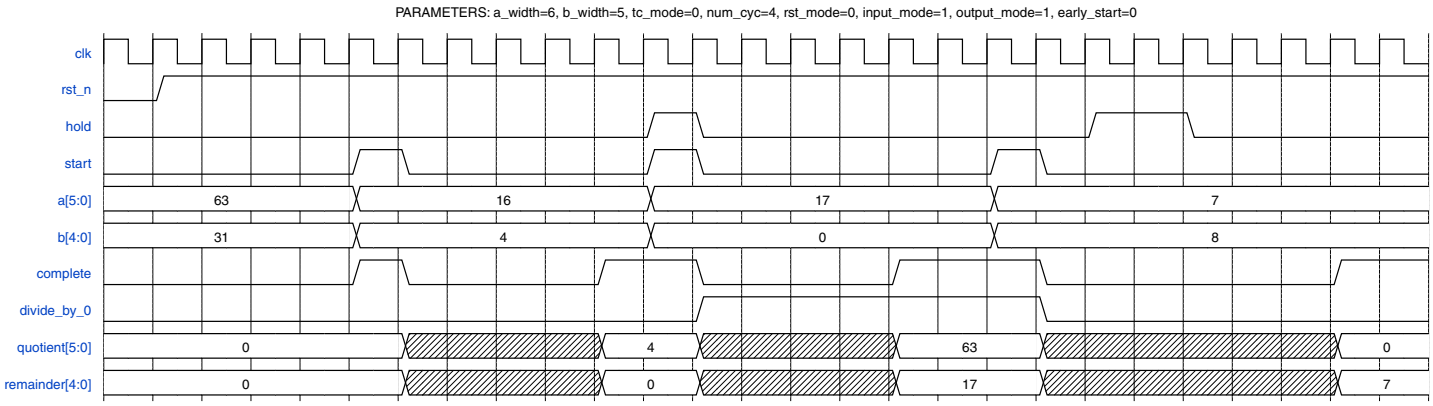
For the parameter settings shown in Figure 1-3, Table 1-6 on page 4 specifies that the result is produced two cycles after `start` goes to 0. With *input_mode* = 1 (registered input), the input operands can be changed after the last sampled `start` = 1 but no sooner, because *early_start* = 1 allows for the processing to begin during the last cycle when `start` = 1. This requires that input operands must be held stable during the last cycle when `start` = 1.

Figure 1-3 Simulation Waveform 3



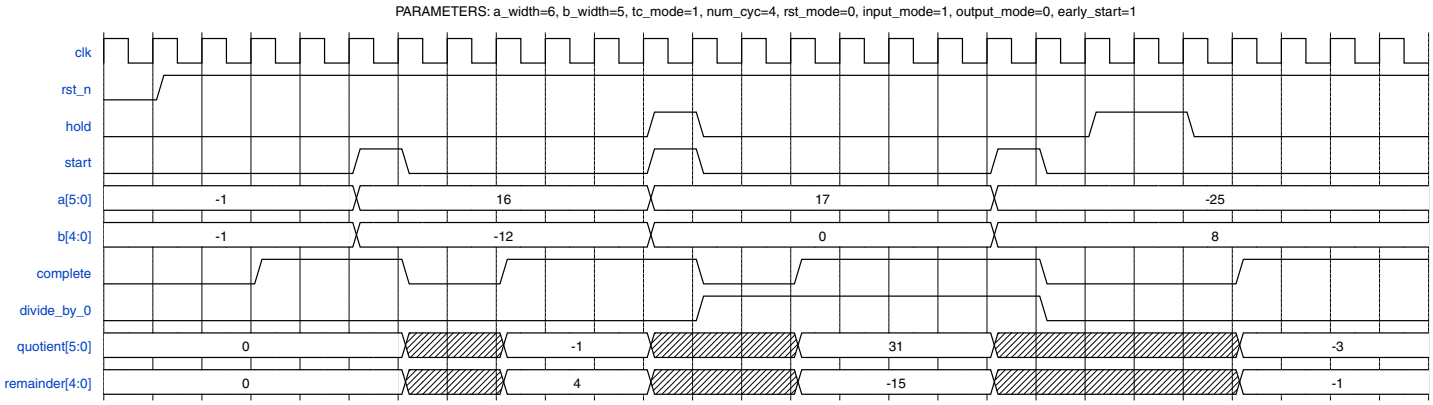
For the parameter settings shown in Figure 1-4, Table 1-6 on page 4 specifies that the result is produced four cycles after start goes to 0. Because *input_mode* = 1 (registered input), the input data can be removed after the first cycle of operation. When `hold`=1 and `start` = 0, the result is delayed by the same number of clock cycles `hold` = 1.

Figure 1-4 Simulation Waveform 4



For the parameter settings shown in Figure 1-5, Table 1-6 on page 4 specifies that the result is produced two clock cycles after start goes to 0. Because *input_mode* = 1 (registered inputs), the input data can be removed after the first cycle of operation. With `hold` = 1 and `start` = 0, the result is delayed by the same number of cycles for which `hold` = 1. Note that the data is registered in the clock cycles when `start` = 1.

Figure 1-5 Simulation Waveform 5



Enabling minPower

You can instantiate this component without enabling minPower, but to achieve power savings from the low-power implementation (at a sub-level--see Table 1-3 on page 3), you must enable minPower optimization, as follows:

- Design Compiler
  - Version P-2019.03 and later:

    ```
    set power_enable_minpower true
    ```

  - Before version P-2019.03 (requires the DesignWare-LP license feature):

    ```
    set synthetic_library {dw_foundation.sldb dw_minpower.sldb}
    set link_library {* $target_library $synthetic_library}
    ```

- Fusion Compiler

  Optimization for minPower is enabled as part of the total_power metric setting. To enable the total_power metric, use the following:

  ```
  set_qor_strategy -stage synthesis -metric total_power
  ```

Related Topics

- Math – Sequential Overview
- DesignWare Building Block IP User Guide

HDL Usage Through Component Instantiation - VHDL

```
library IEEE, DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DW_Foundation_comp_arith.all;

entity DW_div_seq_inst is
  generic (inst_a_width     : POSITIVE := 8; inst_b_width    : POSITIVE := 8;
           inst_tc_mode     : INTEGER := 0;   inst_num_cyc    : INTEGER := 3;
           inst_rst_mode    : INTEGER := 0;   inst_input_mode : INTEGER := 1;
           inst_output_mode : INTEGER := 1;   inst_early_start : INTEGER := 0
           );
   port (inst_clk  : in std_logic;    inst_rst_n : in std_logic;
         inst_hold : in std_logic;    inst_start : in std_logic;
         inst_a    : in std_logic_vector(inst_a_width-1 downto 0);
         inst_b    : in std_logic_vector(inst_b_width-1 downto 0);
         complete_inst : out std_logic;
         divide_by_0_inst : out std_logic;
         quotient_inst    : out std_logic_vector(inst_a_width-1 downto 0);
         remainder_inst   : out std_logic_vector(inst_b_width-1 downto 0)  );
end DW_div_seq_inst;

architecture inst of DW_div_seq_inst is
begin
  -- Instance of DW_div_seq
  U1 : DW_div_seq
    generic map (a_width => inst_a_width,   b_width => inst_b_width,
                 tc_mode => inst_tc_mode,   num_cyc => inst_num_cyc,
                 rst_mode => inst_rst_mode,   input_mode => inst_input_mode,
                 output_mode => inst_output_mode,
                 early_start => inst_early_start   )
  port map (clk => inst_clk,   rst_n => inst_rst_n,
            hold => inst_hold,   start => inst_start,   a => inst_a,
            b => inst_b,   complete => complete_inst,
            divide_by_0 => divide_by_0_inst,   quotient => quotient_inst,
            remainder => remainder_inst   );
end inst;

-- pragma translate_off
configuration DW_div_seq_inst_cfg_inst of DW_div_seq_inst is
  for inst
  end for;
end DW_div_seq_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_div_seq_inst(inst_clk, inst_rst_n, inst_hold, inst_start, inst_a,
   inst_b, complete_inst, divide_by_0_inst, quotient_inst, remainder_inst);

  parameter inst_a_width = 8;
  parameter inst_b_width = 8;
  parameter inst_tc_mode = 0;
  parameter inst_num_cyc = 3;
  parameter inst_rst_mode = 0;
  parameter inst_input_mode = 1;
  parameter inst_output_mode = 1;
  parameter inst_early_start = 0;
  // Please add +incdir+$SYNOPSYS/dw/sim_ver+ to your verilog simulator
  // command line (for simulation).
  input inst_clk;
  input inst_rst_n;
  input inst_hold;
  input inst_start;
  input [inst_a_width-1 : 0] inst_a;
  input [inst_b_width-1 : 0] inst_b;
  output complete_inst;
  output divide_by_0_inst;
  output [inst_a_width-1 : 0] quotient_inst;
  output [inst_b_width-1 : 0] remainder_inst;
  // Instance of DW_div_seq
  DW_div_seq #(inst_a_width, inst_b_width, inst_tc_mode, inst_num_cyc,
               inst_rst_mode, inst_input_mode, inst_output_mode,
               inst_early_start)
    U1 (.clk(inst_clk),   .rst_n(inst_rst_n),   .hold(inst_hold),
        .start(inst_start),   .a(inst_a),   .b(inst_b),
        .complete(complete_inst),   .divide_by_0(divide_by_0_inst),
        .quotient(quotient_inst),   .remainder(remainder_inst) );
endmodule
```

Revision History

For notes about this release, see the *DesignWare Building Block IP Release Notes*.

For lists of both known and fixed issues for this component, refer to the STAR report.

For a version of this datasheet with visible change bars, click here.

Date	Release	Updates
January 2023	DWBB_202212.1	■ Clarified the note in Table 1-6 on page 4 ■ Clarified formulas in "Formula for Dividend Bits Processed Per Cycle" on page 5 and "Formula for Actual Number of Cycles Required" on page 5
July 2020	DWBB_201912.5	■ Adjusted content and title of "Suppressing Warning Messages During Verilog Simulation" on page 5 and added the DW_SUPPRESS_WARN macro and other warning messages
October 2019	DWBB_201903.5	■ Added the "Disabling Clock Monitor Messages" section
March 2019	DWBB_201903.0	■ Clarified license requirements in Table 1-3 on page 3 ■ Added "Enabling minPower" on page 9
August 2017	DWBB_201612.5-1	■ For STARs 9001112685 and 9001226703: - Adjusted the description of computation when *early_start* = 1 on page 4 - Updated the following timing diagrams and the text that describes them: Figure 1-1 on page 7 Figure 1-2 on page 7 Figure 1-3 on page 8 Figure 1-4 on page 8 Figure 1-5 on page 9 ■ Added this Revision History table and the document links on this page

Copyright Notice and Proprietary Information

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.

All other product or company names may be trademarks of their respective owners.

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
www.synopsys.com